IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

HOULIHANE	Atty. Ref.: 550-499
Serial No. 10/743,473	Group: 2863
Filed: December 23, 2003	Examiner: Walling, Meagan S.

For: GENERATION OF A TESTBENCH FOR A REPRESENTATION OF A DEVICE

Before the Board of Patent Appeals and Interferences

BRIEF FOR APPELLANT

On Appeal From Final Rejection
From Group Art Unit 2863

John R. Lastova
NIXON & VANDERHYE P.C.
11th Floor, 901 North Glebe Road
Arlington, Virginia 22203-1808
(703) 816-4025
Attorney for Appellant
Houlihane
ARM Limited

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Patent Application of

HOULIHANE	Atty. Ref.: 550-499
Serial No. 10/743,473	Group: 2863
Filed: December 23, 2003	Examiner: Walling, Meagan S.

For: GENERATION OF A TESTBENCH FOR A REPRESENTATION OF A DEVICE

* *

September 14, 2006

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, ARM Limited, a United Kingdom corporation.

II. RELATED APPEALS AND INTERFERENCES

A first Appeal Brief was filed on March 31, 2006. In response, the final rejection dated November 1, 2005 was withdrawn and a new grounds of rejection set forth in the office action dated June 15, 2006. There are no other appeals related to this subject application. There are no interferences related to this subject application.

III. STATUS OF CLAIMS

Claims 1-46 are pending. Claims 10, 15, 21, 34, 39, and 45 have been indicated as allowable. Claims 1-9, 11-14, 16-20, 22-33, 35-38, 40-44, and 46 stand rejected.

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection or after the new grounds of rejection entered after the prosecution was reopened after Appellant filed its first Appeal Brief.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

When developing components or devices for integration into a system, various test procedures are typically performed to ensure that the device will operate in the desired manner when integrated into the system. The development of a device includes defining a representation of its functional operation and then synthesizing the device representation into a sequence of hardware elements using a synthesizing tool. The result of the synthesis is a device design that can be used to produce the actual hardware device. A testbench model may be used to test the device representation. The testbench model provides a test environment for the device representation and is run on a simulation tool to produce test results which can be analyzed to determine whether the device representation is operating as required.

If a particular instantiation of a configurable device representation is produced based on some specific configuration data, the representation may be placed within a larger system representation that includes representations of other interacting devices. But in this situation, direct control over the input stimuli to the representation of the specific device being tested is lost, resulting in a less than satisfactory test procedure. See page 3, lines 4-10.

As a non-limiting example, the System on Chip (SoC) design 10 in Figure 1 shows devices 15, 20, 25, 35 and 45 interconnected by buses as specified within an interconnect block 60—the device whose design and testing is highlighted in the following example. See page 11, beginning at line 19. The interconnect block 60 includes a bus matrix which provides for the interconnection of multiple bus masters and slaves within the SoC 10. Each master device can access a different subset of the slave devices, and may also use a different memory map. Further, each slave can use one of a number of different arbitration schemes. For any particular instantiation of the interconnect block 60, various parameters will be defined. But in general, there are thousands of possible configurations of the interconnect block. See page 13, line 30-page 14, line 7.

The whole of the SoC design 10 will typically be defined in a Register Transfer Language (RTL) representation, and each device in Figure 1 can be considered as a separate RTL device representation. Each device RTL representation must be tested, either alone or in combination with other device RTL representations, by constructing a testbench around the device representations to provide a test environment for subsequent execution on a simulator tool. This verification process accounts for a significant proportion of the total design effort. See page 13, lines 3-4.

The interconnect block design 60 is intended to be configurable, so that the same basic design can be used in a number of different implementations. But the flexibility in the design of the interconnect block adds to the complexity of verifying the generated design. Given the potentially complex interactions in the design, generating a suitable test sequence to fully exercise a generic interconnect block design is a difficult task, and will not typically verify that all possible instantiations of the interconnect block 60 will always produce a correctly functioning interconnect block. See page 13, lines 9-18.

Although a particular instantiation of the interconnect block 60 can be tested in the context of the SoC design, (see Figure 2), that approach only allows indirect control of the input stimuli to the interconnect block, and therefore, all possibilities of transactions via the various interconnection of buses cannot be tested. See page 3, lines 19-29. The interconnect block 60 could be tested in isolation, as illustrated in Figure 3, with a matching testbench 100 being generated for any particular instantiation of the interconnect block 60. But this approach is normally prohibitive because a specific testbench 100 must be written for each of possibly thousands of instantiations of the interconnect block 60. See page 13, lines 30-page 14, line 4. The inventor conceived of a technique to automatically produce a matching testbench 100 for any particular instantiation of the interconnect block 60. See page 14, lines 4-7.

Figure 4 (reproduced below) shows a processing tool 150 for generating both a representation of a device under test (DUT) 170 (in the above example an interconnect block) and a matching testbench representation 175. A first set of templates 160 is used in generating the testbench 175, and a second set of templates 165 is used in generating the DUT 170. See page 14, lines 8-11.

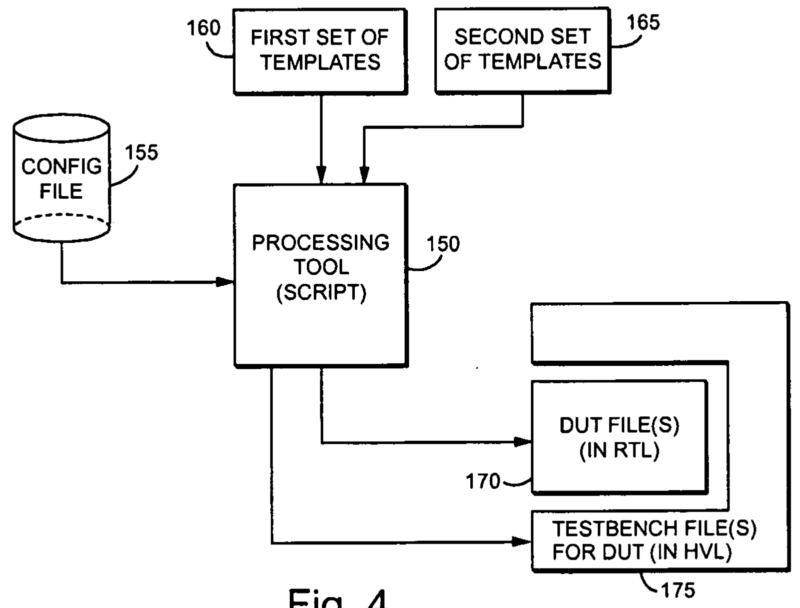

Fig. 4

The second set of templates 165 used by the processing tool to generate the DUT representation 170 includes definitions for each constituent block of that DUT.

The non-limiting example in Figure 5 includes definitions of an input stage, an output stage, a decoder, a multiplexer, etc. For any particular instantiation of the device, the designer specifies a configuration file 155 providing the information required to enable the processing tool to build a representation of that required device from the second set of templates 165. See page 15, lines 1-5.

In one non-limiting example embodiment, the processing tool 150 is a script which processes the constructs provided within the second set of templates with reference to the configuration file 155 to produce one or more files defining the device representation e.g., in RTL. See page 15, lines 9-13. The script 150 also reads a first set of templates 160 and applies the same configuration file 155 to produce one or more output testbench files defining a testbench representation matched to the particular instantiation of the device representation produced using that configuration file 155. An example of the testbench produced for the interconnect block illustrated in Figure 5 is shown in Figure 6 permitting direct control of the testing inputs.

Figure 8 illustrates an example overall process for generating a device representation and a matching testbench representation and then running a simulator to produce test results. See page 19. The processing tool 150 reads the configuration file 155 and generates one or more DUT files with reference to the second set of templates 165. One or more testbench files are generated by the processing tool 150 with reference to the first set of templates 160. When a simulator tool is later run, it reads out the DUT and testbench representation files and performs a simulation to generate some output results.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 23-24 stand rejected under 35 U.S.C. §101. Claims 1-9, 11-14, 16-17, 19-20, 22-23, 35-38, 40-41, 43-44, and 46 stand rejected under 35 U.S.C. §102(e) as being anticipated by Nightingale (US 6,876,941). Claims 1-4, 7, 18, 22-28, 31, 42, and 46 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent Publication U.S. 2004/0111252 to Burgun et al. All rejections are requested for review on appeal.

VII. ARGUMENT

A. Legal Standard for Statutory Subject Matter

The Federal Circuit set forth the legal test to determine whether a claim recites statutory subject matter in the State Street case cited by the Examiner. The claimed subject matter as a whole simply must accomplish a practical application. The court defined practicality as producing a "useful, concrete, and tangible result." State Street Bank & Trust Co. v. Signature Financial Group Inc., 149 Fi.3d 1368, 1373 (Fed. Cir. 1998).

B. <u>Claims 23 and 24 Recite Subject Matter That Is Useful And Which Produces A Concrete And Tangible Result</u>

The Examiner rightly determined that claims 1 and 22 define statutory subject matter. The Board is invited to review claim 1 which recites for example a method like claim 23 and includes the step of "generating the testbench with reference to the configuration data and a first set of templates defining the test environment." It is difficult to reconcile that determination of statutory subject matter of claim 1 with this §101 rejection of claims 23 and 24. But for the following reasons, claims 23 and 24 also recite statutory subject matter.

Claim 23 is directed to a method which is a well-recognized statutory category of invention: "A method of generating a representation of a device to be incorporated in a data processing apparatus and a testbench providing a test environment that represents one or more components of the data processing apparatus with which that device is to be coupled." The method produces something of "real world" value and is more than an idea or concept or a starting point for future investigation or research. Indeed, as explained in the summary above and in more detail in the application, it is a time consuming and expensive process to design a system integrated on a single semiconductor chip and test that design to make sure it works before the integrated circuit is manufactured.

The method recites producing a testbench, which is a software model used to test the device representation. The testbench model is quite useful because it provides a test environment to test a device representation and is executed on a simulation tool to produce test results which can be analyzed to determine whether the device representation is operating as required. Accordingly, the testbench is a concrete "thing"—a tangible, real world tool. Without a test bench, the device design would have to be tested after its built—a risky proposition in the semiconductor manufacturing field.

Claim 23 also recites "employing the processing tool to generate the representation of the device." Again, without the device representation, the actual device would have to be manufactured and then tested in order to see if the device design was suitable for its intended application. If the device turns out to be unsuitable, all that effort and expense is wasted. Like the testbench, the device representation, e.g., defined using register transfer language, is a concrete and tangible "thing" that is useful in device design and testing.

The non-statutory subject matter rejection of claims 23 and 24 should be reversed.

C. Legal Standard for Anticipation

To establish that a claim is anticipated, the Examiner must point out where each and every limitation in the claim is found in a single prior art reference. *Scripps Clinic & Research Found. v. Genentec, Inc.*, 927 F.2d 1565 (Fed. Cir. 1991). Every limitation contained in the claims must be present in the reference, and if even one limitation is missing from the reference, then it does not anticipate the claim. *Kloster Speedsteel AB v. Crucible, Inc.*, 793 F.2d 1565 (Fed. Cir. 1986). Burgun fails to satisfy this exacting standard.

D. <u>Burgun Describes An Emulation/Simulation Platform</u>

In paragraph 0002, Burgun states that his invention applies to "the implementation of a test environment for a circuit whose operation one seeks to validate (and referred to as the 'design under test') and which is *emulated* entirely or partly by a reconfigurable hardware system." Burgun explains that "emulator" means "a reconfigurable hardware system." Paragraph 0003.

Because the test environment embedded in the reconfigurable hardware system and the design under test are both typically implemented by the same configurable resources, Burgun sees a problem with the "very strong dependence between the test environment and the design under test." Paragraph 0017. Burgun's solution is a simulation/emulation platform in which the reconfigurable system emulating the design under test is separated from that emulating the hardware part of the test environment. See paragraphs 0023-25.

E. <u>Burgun Does Not Disclose Generating A Device Representation and A Testbench Representation</u>

Claims 1 and 25, for example, recite:

- "generating a testbench *for* a *representation* of a device to be incorporated in a data processing apparatus,"

- "the *testbench* providing a test environment that *represents* one or more components of the data processing apparatus with which that device is to be coupled,"

- "the *representation* of the device being configurable based on configuration data specifying predetermined attributes of the one or more components."

Claims 23 and 46 similarly recite:

> generating a representation of a device ... and a testbench ... that represents one or more components of the data processing apparatus with which that device is to be coupled, the representation of the device being configurable in dependence on the one or more components.

Burgun lacks these features recited in the independent claims because Burgun is focused on a different phase in device design testing. Burgun starts off with an ***already-generated*** design under test and an ***already-generated*** associated test environment. For example, Burgun states in paragraph 0026: "the invention therefore proposes a method of emulating a design under test associated with a test environment." Paragraph 0076 states that the system in Figure 1 of Burgun "includes an emulator of the design under test EML and an assembly ENVT embodied as a test environment for this same design under test." A design under test, i.e., the representation of a device under test, has already been generated.

There is even an emulator for that design. Similarly, the reconfigurable test bench BTR is part of an already existing "assembly ENVT."

It is in this context that paragraphs 0027-29 must be understood. Assuming an already-generated design under test and an already-generated test environment, Burgun configures reconfigurable hardware circuits to simulate/emulate the operation of that design under test and associated test environment. Thus, Burgun relates to the simulation process described on page 6, lines 12-16.

In contrast, claims 1, 23-25, and 46 are directed to the ***earlier*** phase of generating a representation of the device under test and a representation of its associated test environment, so that those representations can subsequently be simulated using a simulation tool. The Board should consider the non-limiting example sequence of steps shown Figure 8 of the instant application (reproduced below) where steps 605-615 are performed by a processing tool 150 to generate files for a representation of a device under test (DUT) (see step 610) and files for a representation of a test bench file (see step 615) using the *same* configuration data for the DUT representation.

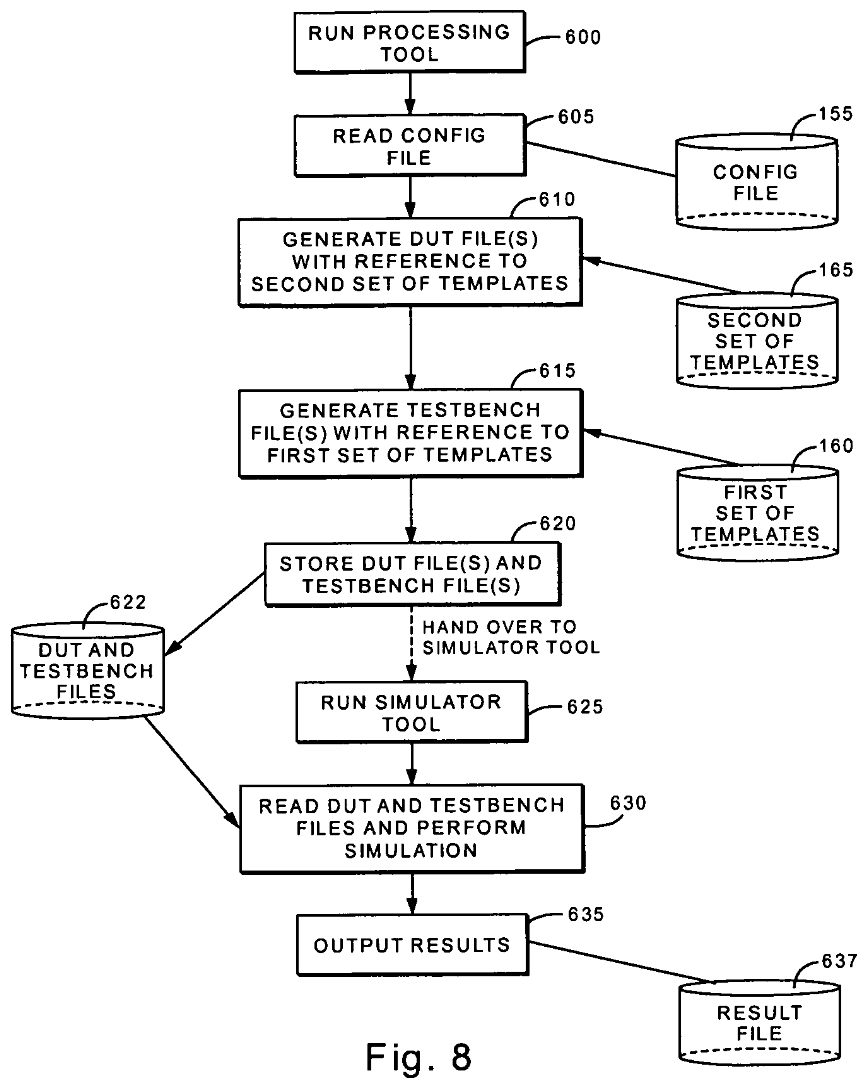
RUN PROCESSING TOOL
600
READ CONFIG FILE
605
CONFIG FILE
155
GENERATE DUT FILE(S) WITH REFERENCE TO SECOND SET OF TEMPLATES
610
SECOND SET OF TEMPLATES
165
GENERATE TESTBENCH FILE(S) WITH REFERENCE TO FIRST SET OF TEMPLATES
615
FIRST SET OF TEMPLATES
160
STORE DUT FILE(S) AND TESTBENCH FILE(S)
620
DUT AND TESTBENCH FILES
622
HAND OVER TO SIMULATOR TOOL
RUN SIMULATOR TOOL
625
READ DUT AND TESTBENCH FILES AND PERFORM SIMULATION
630
OUTPUT RESULTS
635
RESULT FILE
637

Fig. 8

Only after the DUT and testbench representation files have been generated is a simulator tool run (step 625) and simulation performed using those generated file sets (step 630). The *independent* claims are directed to generating these files. Simulation using those files is explicitly recited in *dependent* claims. See for example dependent claim 8. So it is unreasonable to argue that the independent claims can be construed as simulation since a dependent claim adds simulation.

F. <u>Burgun Does Not Disclose Receiving the Configuration Data Used to Configure the Representation of the Device and Generating the Testbench with Reference to the Configuration Data</u>

Claims 1 and 23 recite "receiving the configuration data used to configure the representation of the device" and "generating the testbench with reference to the configuration data and a first set of templates defining the test environment." These features are not described in Burgun.

As demonstrated above, Burgun assumes the existence of the design under test and the test bench BTR. Although both were generated in some fashion, Burgun does not describe how they were generated. There is no disclosure that the test bench BTR was generated based on configuration data used to configure the design under test or a set of templates defining a test environment. The test bench exists. The Examiner fails to point out where Burgun describes ever receiving any kind of data—let alone the claimed configuration data and the first set of templates—used to generate the test bench BTR.

Accordingly, Burgun's configuration files are not used to "configure the representation of the device" (quoted from claim 1) and to generate the associated "testbench with reference to the configuration data and a first set of templates defining the test environment" (quoted from claim 1). Instead, Burgun's configuration files are used to configure the reconfigurable hardware units for emulating an ***already-existing*** design under test and an associated test environment.

G. Burgun Does Not Disclose Generating the Testbench with Reference to the SAME Configuration Data Used to Configure the Representation of the Device

The independent claims require that the test bench be generated with reference to the **same** configuration data used to configure the representation of the device. For example, claim 1 recites at step (b) the testbench is generated with reference to **the** configuration data and a first set of templates. From the earlier recitations in claim 1, **the** configuration data being referred to is the same configuration data used to configure the representation of the device.

This feature of using the same configuration data as claimed is yet another claim feature missing in Burgun. The Examiner relies on paragraphs 0028 and 0032 of Burgun. Paragraphs 0026-30 describe emulating an already-generated design under test with an already-generated test environment (see paragraph 0026). This emulation process involves generating two *different* configuration files. The first configuration file is used to configure a first reconfigurable hardware part that represents the already existing test bench BTR (paragraph 0028). The second configuration file is used to configure a second reconfigurable hardware part EML for emulating the design under test (paragraph 0029). As stressed in paragraph 0029, the first and second reconfigurable hardware parts are "distinct and mutually connected." These two separate parts are provided with their own *separate configuration files* to overcome the problems associated with the "strong dependence" between the test environment and the design under test." See paragraph 0017.

Indeed, Burgun *stresses the importance of using two **different** configuration files* (albeit used for different purposes than generating representations of a device and testbench). The first configuration file configures the reconfigurable hardware representing the testbench and the second configuration file configures the reconfigurable hardware unit acting as an emulator for the device under test. See, for example, Burgun's criticism of a strong dependence between the test environment and the design under test in paragraph 0017 and Burgun's contrasting approach where the two are "separated." Paragraph 0025.

Unlike Burgun, the inventor in this application recognized the value in having the same configuration data be used in generating the representation of the device under test and the representation of the testbench. As a result, it is possible to automatically generate testbench representation files for the device under test (DUT) that match the top-level of the representation of the DUT. This allows any possible configuration of the DUT to be tested when that particular configuration of the DUT has been generated.

H. <u>Burgun Does Not Disclose Generating the Testbench Representation with Reference to Both the Same Configuration Data Used to Configure the Device Representation and a First Set of Templates</u>

The independent claims recite generating "the testbench with reference to the configuration data and a first set of templates defining the test environment." As explained already, Burgun's testbench is assumed to already exist. It has been generated, but Burgun does not explain how. Nor does the Examiner explain where or how Burgun generates the testbench BTR using a "first set of templates defining the test environment."

The Examiner refers to paragraph 0032 in relation to step (b) of claim 1, but does not explain what element in Burgun corresponds to the first set of templates. It is unclear whether the Examiner is taking the position that compilation directives are analogous to the claimed first set of templates. If that is the case, such a position is inconsistent with Burgun's teachings in paragraph 0032 where using the compilation directives results in the first configuration file (see the last line of paragraph 0032). But in claim 1, the first set of templates and the configuration data are used as inputs in generating a new testbench representation rather than configuring an already-existing testbench representation. Paragraph 0032 in Burgun also makes clear that the configuration file is the <u>output</u> of the processing and not an input. Accordingly, the compilation directives being referred to in paragraph 0032 are entirely different than the first set of templates recited in the independent claims.

I. Burgun Does Not Disclose Generating the Device Representation with Reference to Both the Same Configuration Data Used to Configure the Device Representation and a Second Set of Templates

Claims 23 and 46 also recite using "the processing tool to generate the representation of the device with reference to the configuration data and a second set of templates defining the representation of the device." This feature is not disclosed in Burgun. The Examiner fails to explain where or how Burgun generates a representation of the device under test using "second set of templates defining the representation of the device." Burgun's device representation has already been generated, but Burgun does not say how. Nor does the Examiner specifically identify what element in Burgun supposedly corresponds to the claimed second set of templates.

J. Nightingale Does Not Generate The Representation Of The Device Under Test (DUT)

Nightingale assumes that the representation of the device under test (DUT) already exists and focuses on testing compliance of that DUT with a bus protocol: "the user preferably supplies the user defined test sequence 305, an RTL implementation of the DUT 330, and a user configuration file providing configuration information specific to the DUT." Col. 11, lines 5-8. The DUT is already defined. The configuration file provides additional information used for creating a test environment for the DUT, but that configuration file is not used to create the DUT RTL implementation. See also Figure 6 in which an initial step

610 loads a DUT interface module in which "the RTL representation of the DUT is instantiated." Col. 14, line 5. Figure 7 further includes an initial step 640: "generate compliance test environment for DUT."

Accordingly, the DUT representation file has already been created in Nightingale. In contrast, claims 2 and 23 recite "employing the processing tool to generate the representation of the device with reference to the configuration data and a second set of templates defining the representation of the device." Claims 26 and 46 have similar apparatus language. The Examiner refers to col. 3, lines 15-20 and col. 6, line 49. These sections do not describe a second set of templates. Nor do they refer to the device to be tested being configurable in any respect.

J. <u>Nightingale Does Not Configure The Representation Of The Device Under Test (DUT) Using The Claimed Configuration Data</u>

In all the independent claims, the claimed DUT representation is "configurable in dependence on the one or more components" of the "data processing apparatus with which that device is to be coupled." The independent claims describe a configurable representation of a device to be tested.

In contrast, Nightingale's DUT has already been defined or implemented as explained above. The configuration file described in Nightingale identifies the type of the device and capabilities of the DUT. It is not used to configure the DUT representation. Examples given of the device type include master, slave, arbiter, decoder, etc., and examples of the device capabilities are a subset of a bus protocol that the device needs to support. This configuration data limits the amount of testing so that "it is not necessary to test for compliance with parts of the bus protocol not used by the device." Col. 4, lines 15-16.

Nightingale's configuration file is for generating test environment and not for configuring the DUT representation. A configuration engine in Nightingale dynamically generates a test environment for that device based on that configuration file. Column 3, lines 15-20. The configuration file identifies the type and capabilities of the device to be tested and is used to decide which test components to use to test the device. See col. 3, lines 13-15. Nightingale's configuration file is only produced once the device to be tested has been selected—it is not used to configure the DUT representation.

K. Nightingale Does Not Describe Generate The DUT Testbench With Reference To The Configuration Data And A First Set Of Templates Defining The Test Environment

The same configuration data used to configure that representation of the device is also used to generate the testbench for that device representation. The Examiner refers to column 3, lines 15 to 20 in Nightingale, which describes generating a test environment by creating test components which are selected depending on a configuration file. As explained earlier, this configuration file is not the **same** configuration data recited in the claims that is used to configure the device representation to be tested.

Because the same configuration data that is used to configure the representation of the device is also used to generate a testbench, it is possible to generate a testbench which matches the top-level of the DUT. As a result, any possible configuration of the DUT to be tested is allowed once that particular configuration of the DUT has been generated. A matching testbench can be generated for any particular instantiation of any customized design, which allows that customized design to be tested in isolation using a comprehensive verification test. In particular, direct control of the input stimuli to that DUT is possible.

Nightingale also lacks the claimed first set of templates defining the test environment. The Examiner reads this claim feature onto Nightingale's testbench 300 using the configuration file. This reading is wrong.

Nightingale's configuration engine defines the test environment. For any particular test environment instantiation, the contents of user-provided configuration file determines which test components to include in the test environment. See col. 3, lines 15-20 and 39-54. In contrast, the claimed first set of templates defines the test environment, and the testbench is generated with reference to the first set of templates **and** the **same** configuration data that is used to configure the representation of the device. These features are lacking in Nightingale.

The only reference to "templates" in Nightingale is in the context of an embodiment where the configuration file is selected from a set of configuration file templates. There is "a configuration file template for each type of device," see col. 4, lines 3-8, that define the device type and capabilities. Col. 4, lines 9-13. These templates in Nightingale do not "define the test environment" as recited in the independent claims.

VIII. CONCLUSION

The non-statutory subject matter rejection is based on a faulty premise that requires an outputting, displaying, or storing step. The *State Street* standard does not impose that requirement. Claims 23 and 24 have real world utility and produce useful, concrete, and tangible results. The Board should reverse the non-statutory subject matter rejection.

The anticipation rejections are also contrary to law. As clearly demonstrated above, there are multiple phrases recited in the independent claims that simply are not disclosed in Burgun and Nightingale. Simply stating that the claims are "broader" that what is described in the specification is not enough to demonstrate anticipation. Rather, the Examiner must identify where and explain how each claim feature is described in Burgun and Nightingale. The Examiner fails to meet that burden. Given the fact that multiple claim features as argued above are missing from Burgun, the Board should reverse the outstanding anticipation rejections.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: [illegible]
John R. Lastova
Reg. No. 33,149

JRL/sd
Appendix A - Claims on Appeal

IX. CLAIMS APPENDIX

1. A method of generating a testbench for a representation of a device to be incorporated in a data processing apparatus, the testbench providing a test environment that represents one or more components of the data processing apparatus with which that device is to be coupled, the representation of the device being configurable based on configuration data specifying predetermined attributes of the one or more components, the method comprising the steps of:

(a) receiving the configuration data used to configure the representation of the device; and

(b) generating the testbench with reference to the configuration data and a first set of templates defining the test environment.

2. A method as claimed in Claim 1, further comprising the step of:

(c) generating the representation of the device with reference to the configuration data and a second set of templates defining the representation of the device.

3. A method as claimed in Claim 2, further comprising the step of:
providing a processing tool having access to the configuration data and the first and second sets of templates, said steps (b) and (c) being performed by the processing tool.

4. A method as claimed in Claim 3, wherein the processing tool is operable independent of a language produced by the processing tool from each template.

5. A method as claimed in Claim 1, wherein the representation of the device is provided in a first language type and at said step (b) a part of the testbench defined by a number of the templates in the first set is generated in a second language type different to the first language type.

6. A method as claimed in Claim 5, wherein said first language type is a Register Transfer Language (RTL), and said second language type is a High level Verification Language (HVL).

7. A method as claimed in Claim 1, wherein said device is a bus interconnect block.

8. A method as claimed in Claim 7, further comprising the step of:

employing a simulation tool to run a model of the data processing apparatus using the representation of the device and the testbench;

wherein the first set of templates includes a master template defining a master engine coupled to a bus and operable during running of the model to generate test stimuli for input via the bus to the representation of the device.

9. A method as claimed in Claim 8, wherein the master template includes a master monitor operable during running of the model to monitor signals on the bus to which the master engine is coupled.

10. A method as claimed in Claim 9, wherein the testbench includes a scoreboard for checking data integrity within the model, and the master monitor is operable to output data to the scoreboard indicative of the signals on the bus to which the master engine is coupled.

11. A method as claimed in Claim 8, wherein the master template includes a checker operable during running of the model to check that signals at an interface between the master engine and the bus to which the master engine is coupled conform to a protocol for that bus.

12. A method as claimed in Claim 8, wherein the master engine is arranged to generate the test stimuli in a random manner.

13. A method as claimed in Claim 7, further comprising the step of:

employing a simulation tool to run a model of the data processing apparatus using the representation of the device and the testbench;

wherein the first set of templates includes a slave template defining a slave engine coupled to a bus and operable during running of the model to generate response signals in reply to test stimuli received from the representation of the device.

14. A method as claimed in Claim 13, wherein the slave template includes a slave monitor operable during running of the model to monitor signals on the bus to which the slave engine is coupled.

15. A method as claimed in Claim 14, wherein the testbench includes a scoreboard for checking data integrity within the model, and the slave monitor is operable to output data to the scoreboard indicative of the signals on the bus to which the slave engine is coupled.

16. A method as claimed in Claim 13, wherein the slave template includes a checker operable during running of the model to check that signals at an interface between the slave engine and the bus to which the slave engine is coupled conform to a protocol for that bus.

17. A method as claimed in Claim 13, wherein the slave engine is arranged to generate the response signals in a random manner.

18. A method as claimed in Claim 2, wherein the representation of the device is formed from constituent blocks and the second set of templates defines the representation of the device and its constituent blocks.

19. A method as claimed in Claim 1, wherein there are a number of different component types, and the predetermined attributes specified by the configuration data indicate the component type for each of said one or more components.

20. A method as claimed in Claim 19, wherein said device is a bus interconnect block, and wherein one of the component types is a master type, and for each of said one or more components which is a master type, the predetermined attributes identify connections to any slave components within said one or more components that that master type component is connected to.

21. A method as claimed in Claim 20, wherein the connections are identified as either connections to a local slave component not shared with other master components, or as connections through a bus matrix of the bus interconnect block to a shared slave component shared with one or more other master components.

22. A computer program product comprising code portions operable to control a computer to perform a method as claimed in Claim 1.

23. A method of generating a representation of a device to be incorporated in a data processing apparatus and a testbench providing a test environment that represents one or more components of the data processing apparatus with which that device is to be coupled, the representation of the device being configurable in dependence on the one or more components, the method comprising the steps of:
(a) receiving a configuration data specifying predetermined attributes of the one or more components;
(b) employing a processing tool to generate the testbench with reference to the configuration data and a first set of templates defining the test environment; and

(c) employing the processing tool to generate the representation of the device with reference to the configuration data and a second set of templates defining the representation of the device.

24. A computer program product comprising code portions operable to control a computer to perform a method as claimed in Claim 23.

25. A system for generating a testbench for a representation of a device to be incorporated in a data processing apparatus, the testbench providing a test environment that represents one or more components of the data processing apparatus with which that device is to be coupled, the representation of the device being configurable based on configuration data specifying predetermined attributes of the one or more components, the system comprising:

logic operable to read the configuration data used to configure the representation of the device; and

generation logic operable to generate the testbench with reference to the configuration data and a first set of templates defining the test environment.

26. A system as claimed in Claim 25, wherein the generation logic is further operable to generate the representation of the device with reference to the configuration data and a second set of templates defining the representation of the device.

27. A system as claimed in Claim 26, further comprising:

a processing tool having access to the configuration data and the first and second sets of templates, the generation logic being provided by the processing tool.

28. A system as claimed in Claim 27, wherein the processing tool is operable independent of a language produced by the processing tool from each template.

29. A system as claimed in Claim 25, wherein the representation of the device is provided in a first language type, and during generation of the testbench by the generation logic a part of the testbench defined by a number of the templates in the first set is generated in a second language type different to the first language type.

30. A system as claimed in Claim 29, wherein said first language type is a Register Transfer Language (RTL), and said second language type is a High level Verification Language (HVL).

31. A system as claimed in Claim 25, wherein said device is a bus interconnect block.

32. A system as claimed in Claim 31, further comprising:

a simulation tool operable to run a model of the data processing apparatus using the representation of the device and the testbench;

wherein the first set of templates includes a master template defining a master engine coupled to a bus and operable during running of the model to generate test stimuli for input via the bus to the representation of the device.

33. A system as claimed in Claim 32, wherein the master template includes a master monitor operable during running of the model to monitor signals on the bus to which the master engine is coupled.

34. A system as claimed in Claim 33, wherein the testbench includes a scoreboard for checking data integrity within the model, and the master monitor is operable to output data to the scoreboard indicative of the signals on the bus to which the master engine is coupled.

35. A system as claimed in Claim 32, wherein the master template includes a checker operable during running of the model to check that signals at an interface between the master engine and the bus to which the master engine is coupled conform to a protocol for that bus.

36. A system as claimed in Claim 32, wherein the master engine is arranged to generate the test stimuli in a random manner.

37. A system as claimed in Claim 31, further comprising:

a simulation tool operable to run a model of the data processing apparatus using the representation of the device and the testbench;

wherein the first set of templates includes a slave template defining a slave engine coupled to a bus and operable during running of the model to generate response signals in reply to test stimuli received from the representation of the device.

38. A system as claimed in Claim 37, wherein the slave template includes a slave monitor operable during running of the model to monitor signals on the bus to which the slave engine is coupled.

39. A system as claimed in Claim 38, wherein the testbench includes a scoreboard for checking data integrity within the model, and the slave monitor is operable to output data to the scoreboard indicative of the signals on the bus to which the slave engine is coupled.

40. A system as claimed in Claim 37, wherein the slave template includes a checker operable during running of the model to check that signals at an interface between the slave engine and the bus to which the slave engine is coupled conform to a protocol for that bus.

41. A system as claimed in Claim 37, wherein the slave engine is arranged to generate the response signals in a random manner.

42. A system as claimed in Claim 26, wherein the representation of the device is formed from constituent blocks and the second set of templates defines the representation of the device and its constituent blocks.

43. A system as claimed in Claim 25, wherein there are a number of different component types, and the predetermined attributes specified by the configuration data indicate the component type for each of said one or more components.

44. A system as claimed in Claim 43, wherein said device is a bus interconnect block, and wherein one of the component types is a master type, and for each of said one or more components which is a master type, the predetermined attributes identify connections to any slave components within said one or more components that that master type component is connected to.

45. A system as claimed in Claim 44, wherein the connections are identified as either connections to a local slave component not shared with other master components, or as connections through a bus matrix of the bus interconnect block to a shared slave component shared with one or more other master components.

46. A system for generating a representation of a device to be incorporated in a data processing apparatus and a testbench providing a test environment that represents one or more components of the data processing apparatus with which that device is to be coupled, the representation of the device being configurable in dependence on the one or more components, the system comprising:

logic operable to read a configuration data specifying predetermined attributes of the one or more components;

a processing tool operable to generate the testbench with reference to the configuration data and a first set of templates defining the test environment; and

the processing tool further being operable to generate the representation of the device with reference to the configuration data and a second set of templates defining the representation of the device.

X. EVIDENCE APPENDIX

There is no evidence appendix.

XI. RELATED PROCEEDINGS APPENDIX

There is no related proceedings appendix.